智能语音客服

目录

一、 系统架构.....3
二、系统实现.....3
1、呼叫中心服务搭建.....3
1.1freeswitch 的搭建（呼叫中心的搭建）.....3
1.2 遇到的问题.....4
1.3freeswitch 的各种模块的装载.....6
2、 Mrcp 的搭建基于百度的 tts+asr.....8
2.1 环境需求.....8
2.2 搭建 MrcpServer.....9
2.3 初始化成功.....10
2.4 使用 mrcpserver.....10
2.5、服务启动中遇到的问题.....11
2.6 配置外网 ip、修改端口号.....12
2.7Freeswitch 安装 mod_unimrcp 模块.....13
3、 话术业务逻辑的实现.....14
话术文件设置如下：.....14
4、 呼叫中心的实现.....16
1）设置呼叫计划.....16
2）设置座席队列.....16
5、 后台业务逻辑的实现.....17
5.1)用户的注册登录.....17
5.2)用户的通话记录.....17
5.3)话术自动拨打.....17
6、 网页电话 webrtc 的实现.....17
7、 前端页面的实现.....18
1、登录注册.....18
2、 电话记录管理.....19
3、webrtc 网页电话.....20
4、 订单话术.....20

	需求分析	架构设计	Freeswitch+mrcp+百度 tts/asr	话术处理	ESL 开发	后台开发+esl 合成	Webrtc	前端展示
朱明龙	√	√		√	√			
周家桢	√		√					
陈桢秀	√	√				√		
王建强	√					√	√	
范苏东	√						√	√

一、系统架构

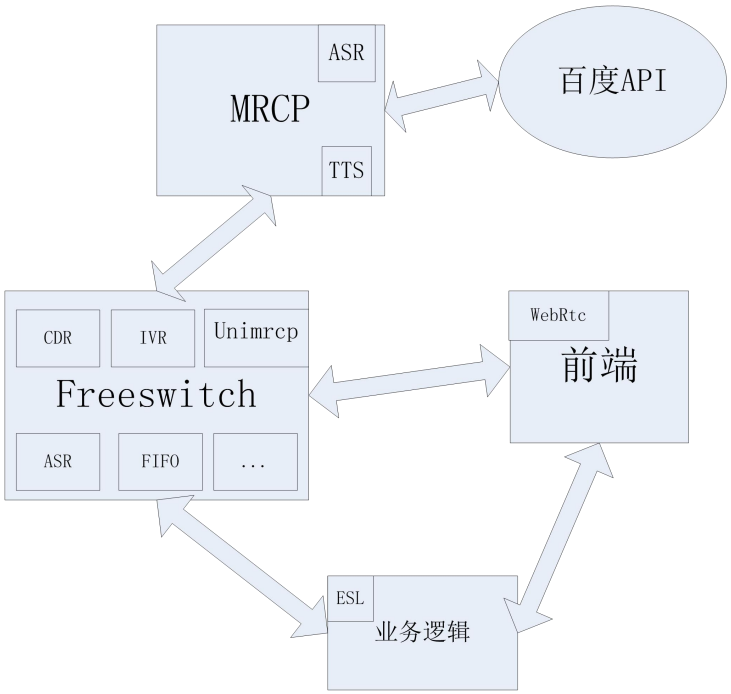


图 1、系统模块图

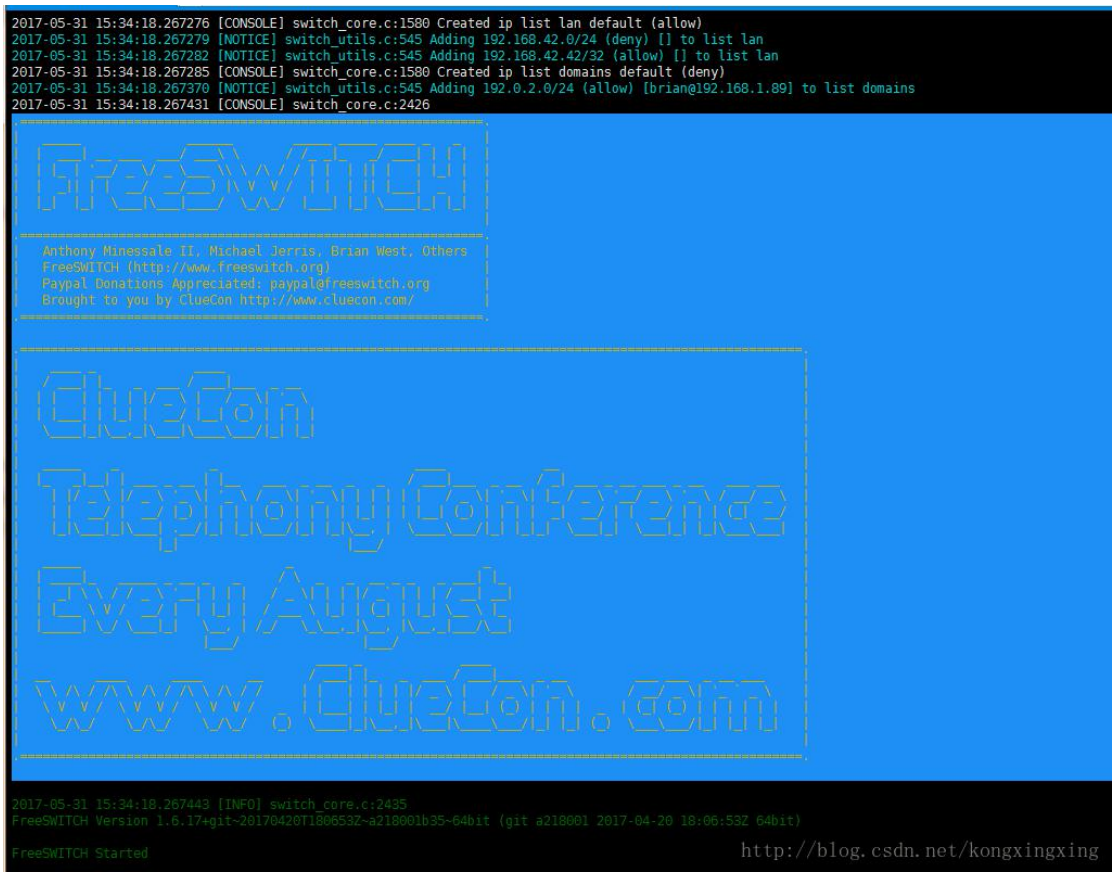
二、系统实现

1、呼叫中心服务搭建

1.1freeswitch 的搭建（呼叫中心的搭建）

下载 git sudo apt-get install git
然后使用 git 下载 freeswitch git clone(要记得注意版本)
https://freeswitch.org/stash/scm/fs/freeswitch.git
由于在 ubuntu 系统中所以不用 homebrew 而是直接使用
apt-get installlibedit-dev libldns-dev
libpcre3-dev libspeexdsp-dev libspeex-dev libcurl4-openssl-dev libopus-dev
libncurses5-dev libtiff-dev libjpeg-dev zlib1g-dev libssl-dev libsqlite3-dev
build-essential automake autoconf git-core wget libtool

liblua50-dev libsndfile1-dev yasm
sudo apt-get install gawk
update-alternatives –set awk /usr/bin/gawk
cd /usr/local/freeswitch 文件夹下
./bootstrap.sh
./configure
make
sudo make install
sudo make uhd-sounds-install
sudo make uhd-moh-install
sudo make install
sudo make uhd-sounds-install
sudo make uhd-moh-install
建立软链接　(目的是随处可使用命令)
ln　–sf /usr/local/freeswitch/bin/freeswitch /usr/local/bin/
ln –sf /usr/local/freeswitch/bin/fs_cli /usr/local/bin
su　输入密码
./freeswitch　启动服务



1.2 遇到的问题

1)、./bootstrap.sh 后如果出现错误，bootstrap: libtool not found.

You need libtool version 1.5.14 or newer to build FreeSWITCH from source.

在 ubuntu 只有 libtoolize，修改 bootstrap.sh，

libtool=${LIBTOOL:-`${LIBDIR}/apr/build/PrintPath glibtool libtool libtool22 libtool15 libtool14 libtoolize`}

2)、./bootstrap.sh 会出现权限的的问题，这时候就需要修改文件夹的权限

chmod 777 对应报错文件 或者 chmod a+x 对应报错文件

3)、lua 的安装问题，还有链接库没有的问题

出现 can not find -llua 是因为找不到 liblua.so，需创建软链接

ln liblua50.so liblua.so

解决方案：

sudo apt-get install liblua5.3-dev

sudo find –name “liblua.so”

ln –sf liblua5.3.so liblua.so

4)、Ubuntu 的启动问题，在正常启动情况下总是会出现如下的问题

Cannot open pid file /usr/local/freeswitch/run/freeswitch.pid

Cannot lock pid file /user/local/freeswitch/run/freeswitch.pid

这是由于权限不够造成的

解决方法：passwd su

输入新的密码

然后登陆 su，输入密码启动 freeswitch

./freeswitch

也有人说是端口占用造成的，这时候便使用(在上面进入 su 的情况下)

netstat –ap|grep 8021/708

Kill -9 pid

1.3freeswitch 的各种模块的装载

整个项目需要多个模块的配合，并且很多模块也是使用时才知道需要载入，所以这边只是必须的模块，并非是全部模块，有 ivr 模块，fifo 模块，callcenter 模块，shout 模块，rtc 模块，cdr 模块，socket 模块等。

1）集成 tts 模块

tts 使用的方法，由于百度 mrcp 没有集成 tts 功能的实际例子，因此，这边我也不知道如何去使用它，所以使用了两种方法去使用它

具体方法

方法 1：使用 playback 的 app 方法

<action application="playback"
data="shout://tsn.baidu.com/text2audio?lan=zh&ctp=1&cuid=abcdxxx&tok=24.4b54bfeae948c983f2d9b3ad19277cf4.2592000.1542251039.282335-14434229&tex=%E7%99%BE%E5%BA%A6%E4%BD%A0%E5%A5%BD%E5%93%88%E5%93%88%E5%93%88%E5%93%88%E5%93%88%E5%93%88&vol=9&per=0&spd=5&pit=5&aue=3"/>

方法 2: 加载 tts_commandline 的情况下配置 tts_commandline.conf.xml 文件中
进行配置，然后在 default.xml 文件中进行配置

方法 3:lua 中进行使用 session 的方法（在后面 1 编写的 esl 中也是使用该 app）

session:execute("playback",
"shout://tsn.baidu.com/text2audio?lan=zh&ctp=1&cuid=abcdxxx&tok=24.4b54bfeae948c983f2d9b3ad19277cf4.2592000.1542251039.282335-14434229&tex=%E7%99%BE%E5%BA%A6%E4%BD%A0%E5%A5%BD%E5%93%88%E5%93%88%E5%93%88%E5%93%88%E5%93%88%E5%93%88&vol=9&per=0&spd=5&pit=5&aue=3")

方法 4：使用 speak 的 app

result = session:execute("speak","content...");

看到这里可以发现所有的 diaplan 中的 app 都可以通过 session:execute(“ ”,” ”);
来实现。

方法 5：使用 esl 来调用 speak 中的 app（项目使用该方法）

2）、集成 ASR

2.1)、出现问题 recognizer channel error

Make sure the vasre-mrcp server is running and is listening on 192.168.1.79:1554. When I installed, it was listening on 127.0.0.1 and I had to change the config in /opt/Vestec/mrcp/conf/mrcpserver.xml.

If iptables is running, make sure incoming connections to port 1554 is allowed (or shut it off) and RTP can be sent to the port range defined in the mrcpserver.xml file.

/user/local/freeswitch/conf/mrcp_profiles.xml

里面的端口以及 ip 地址要和 unimrcpserver.xml 对应，我错误的原因

一开始 rtpd 的 ip 地址写错了，其次两个文件中的端口不一致

<param name=”rtp-port-min” value=”4000”>

<param name=”rtp-port-max” value=”5000”>

2.2）、报错 lua 的 attempt to call field “?”什么问题

这个问题不是 lua 语言的问题，而是自己的文件配置的有问题，配置不对导致无法找到文件。

上面的 rtp 的 ip 配置出现问题。

3）WebRTC 配置

修改 vars.xml，找到 global_codec_prefs，添加 VP8 的支持：

<X-PRE-PROCESS cmd="set" data="global_codec_prefs=G722,H264,PCMU,PCMA,VP8"/>

<X-PRE-PROCESS cmd="set" data="outbound_codec_prefs=G722,H264,PCMU,PCMA,VP8"/>

在 internal.xml 中打开 wss 绑定：

<!-- for sip over secure websocket support -->

<!-- You need wss.pem in $${certs_dir} for wss or one will be created for you -->

<param name="wss-binding" value=":7443"/>

在 conf/autoload_configs/acl.conf.xml 中添加

<list name="wan.auto" default="allow">

<node type="allow" cidr="XXX.XXX.XXX.0/24"/></list>

XXX.XXX.XXX 换成自己的网段

conf/sip_profiles/internal.xml 加上：

<param name="apply-candidate-acl" value="wan.auto"/>

conf/sip_profiles/internal.xml 打开：

<param name="ws-binding" value=":5066"/>

vars.xml 设置：

<X-PRE-PROCESS cmd="set" data="internal_ssl_enable=true"/>

<X-PRE-PROCESS cmd="set" data="external_auth_calls=true"/>

<X-PRE-PROCESS cmd="set" data="external_ssl_enable=true"/>

Websocket：Socket connect refused.

首先要明白 inbound 是从 freeswitch 到 esl 客户端，配置 outbound 的时候，里面的 socket 是 esl 的地址和端口。出现 socket connect refused 的原因是因为 windows 下防火墙没有关闭。

mod_shout 模块

该模块允许以任何采样率播放本地和远程 mp3 文件。

在构建中启用模块。

编辑源目录中的 modules.conf 以添加 mod_shout。

formats/mod_shout

安装模块：

sudo make mod_shout-install

设置默认启动：

在 freeswitch 中加载模块 load mod_shout

设置默认启动

vim /usr/local/freeswitch/conf/autoload_configs/modules.conf.xml

<load module="mod_shout"/>

2、Mrcp 的搭建基于百度的 tts+asr

2.1 环境需求

Linux 64 位 centos 6u3

gcc: 482 以上 libc，CXX11，百度提供 gcc4.8.2 压缩包，存放于 libs 目录下，解压后执行

bootstrap.sh 完成默认配置;yum install gcc

curl : 7.33，安装方法 yum install curl

ssl : 1.0.1i，安装方法 yum install mod_ssl

daemontools 守护进程（不要忘记这个）

须有 root 账户权限运行，curl、ssl 服务器未安装或版本不够，请自行安装或升级。

2.2 搭建 MrcpServer

cd ~# 下载 MrcpServer 安装包

wget http://tianzhi-public.bj.bcebos.com/MrcpServerV1.2.tar.gz

#解压安装包

tar xvzf MrcpServerV1.2.tar.gz

前往解压后的安装包

cd unimrcp

编译安装

sh bootstrap.sh

如果安装失败，提示请切换至 root 账号，

手动执行 ln -s /root/unimrcp/libs/gcc482 /opt/compiler/gcc-4.8.2，

程序将自动查找/opt/compiler/gcc-4.8.2 的 gcc,可以按照一下方法：

cd ${SERVER_ROOT}/unimrcp/libs

解压安装 gcc

tar xvzf /root/unimrcp/libs/gcc482.tar.gz

创建/opt/compiler/gcc-4.8.2 目录

mkdir -p /opt/compiler/gcc-4.8.2

创建 gcc-4.8.2 软链接

sudo ln -s /root/unimrcp/libs/gcc482 /opt/compiler/gcc-4.8.2

重新编译安装

cd ..

sh bootstrap.sh

如果出现以下提示，表示安装成功：

bootstrap: 使用百度自带 gcc4.8.2 初始化客户环境

2.3 初始化成功

建议设置 contab 定时任务拆分日志:[1 * * * * sh /root/unimrcp/bin/splitLog.sh]

参数配置

从官方文档啊我们可以看出目录结构，只需在${SERVER_ROOT}/conf/recogplugin.json 修改 API Key 和 Secret Key。

其他参数，无特殊需求，无需修改，保持现状。

"app.appKey": "API Key", "app.appSecret": "Secret Key",

2.4 使用 mrcpserver

每次替换 MrcpServer 安装包，都需要在${SERVER_ROOT}/目录，root 权限下执行 sh bootstrap.sh，主要功能：完成百度自带 gcc4.8.2 的环境配置。

启动： 在${SERVER_ROOT}/bin 目录执行

./control start

停止： 在${SERVER_ROOT}/bin 目录执行

./control stop

重启： 在${SERVER_ROOT}/bin 目录执行

./control restart

查看服务状态：在${SERVER_ROOT}/bin 目录执行

./control status

2.5、服务启动中遇到的问题

若执行 control start 失败，请按照以下步骤排查服务

1）、在${SERVER_ROOT}/bin 目录执行

[root@xxx bin]$ ./unimrcpserver bash: ./unimrcpserver: /opt/compiler/gcc-4.8.2/lib64/ld-linux-x86-64.so.2: bad ELF interpreter: No such file or directory，

说明 bootstrap.sh 执行失败，请移步压缩包自带 README 文件，手动完成 gcc4.8.2 的配置。

2）、若单独启动 unimrcpserver 正常，命令行直接输入 supervise，若 supervise 非系统命令，说明守护进程安装失败，请检查守护进程,安装教程如下：

wget --no-check-certificate http://cr.yp.to/daemontools/daemontools-0.76.tar.gz

tar zxf daemontools-0.76.tar.gz

cd admin/daemontools-0.76/

sed -i 's/extern int errno;/#include <errno.h>/1' ./src/error.h

sudo ./package/install

启动 daemontools 工具,&表示后台运行 ：

/command/svscanboot &

然后运行 ps -ef | grep svs 查看运行状态：

root 23697 20807 0 11:36 pts/0 00:00:00 /bin/sh /command/svscanboot

root 23699 23697 0 11:36 pts/0 00:00:00 svscan /service

root 23702 20807 0 11:36 pts/0 00:00:00 grep --color=auto svs

参考 daemontools 的安装、简介

2.6 配置外网 ip、修改端口号

conf/unimrcpserver.xml 配置中配置外网 ip 方法，配置并打开以下参数注释

<sip-uas id="SIP-Agent-1" type="SofiaSIP">

<sip-ip>本机内网地址</sip-ip>

<sip-ext-ip>本机外网地址</sip-ext-ip></sip-uas>

sip 端口修改：

<sip-port>8060</sip-port>替换成设置参数

rtp ip&端口范围修改：<rtp-factory id="RTP-Factory-1">

<rtp-ip>10.10.0.1</rtp-ip> 本机内网 ip

<rtp-ext-ip>a.b.c.d</rtp-ext-ip> 本机外网 ip

<rtp-port-min>5000</rtp-port-min> rtp 端口下限

<rtp-port-max>6000</rtp-port-max> rtp 端口上限</rtp-factory>

2.7Freeswitch 安装 mod_unimrcp 模块

1）、安装 mod_unimrcp 模块
cd /项目源码地址/frerswitch
vim modules.conf

取消掉 asr_tts/mod_unimrcp 的注释
asr_tts/mod_unimrcp

安装 mod_unimrcp 模块
make mod_unimrcp-install

编辑/usr/local/freeswitch/conf/autoload_configs/modules.conf.xml，添加或者去掉注释 mod_unimrcp，让模块启动默认加载
vim /usr/local/freeswitch/conf/autoload_configs/modules.conf.xml

<load module="mod_unimrcp">

2）、设置 profile 文件与 conf 文件；
在 mrcp_profiles 目录下新建 unimrcpserver-mrcp-v2.xml 配置文件：
vim /usr/local/freeswitch/conf/mrcp_profiles/unimrcpserver-mrcp-v2.xml
然后输入以下内容:

```
<include>
    <!-- UniMRCP Server MRCPv2 -->
    <!-- 后面我们使用该配置文件，均使用 name 作为唯一标识，而不是文件名 -->
    <profile name="unimrcpserver-mrcp2" version="2">
    <!-- MRCP 服务器地址和 SIP 端口号 -->
    <param name="server-ip" value="192.168.16.4"/>
    <param name="server-port" value="8060"/>
    <param name="resource-location" value=""/>

    <!-- FreeSWITCH IP、端口以及 SIP 传输方式 -->
    <param name="client-ip" value="192.168.16.4" />
    <param name="client-port" value="5069"/>
    <param name="sip-transport" value="udp"/>

    <param name="speechsynth" value="speechsynthesizer"/>
    <param name="speechrecog" value="speechrecognizer"/>
    <!--param name="rtp-ext-ip" value="auto"/-->
    <param name="rtp-ip" value="192.168.16.4"/>
    <param name="rtp-port-min" value="4000"/>
    <param name="rtp-port-max" value="5000"/>
    <param name="codecs" value="PCMU PCMA L16/96/8000"/>
```

```
<!-- Add any default MRCP params for SPEAK requests here -->
<synthparams>
</synthparams>

<!-- Add any default MRCP params for RECOGNIZE requests here -->
<recogparams>
  <!--param name="start-input-timers" value="false"/-->
</recogparams>
</profile></include>
```

注意，server-ip 为你上一篇文章你部署的搭建百度 Mrcp Server 与 Freeswitch 的 mod_unimrcp 对接实现智能客服的 IP 地址,你可以使用 ifconfig 查看内网 IP 地址。

接下来修改 unimrcp 默认使用的 ASR 驱动，可以使用 vim /usr/local/freeswitch/conf/autoload_configs/unimrcp.conf.xml 编辑修改 default-tts-profile 和 default-asr-profile 为我们新创建的 unimrcpserver-mrcp2：

<!-- UniMRCP profile to use for TTS --><param name="default-tts-profile" value="unimrcpserver-mrcp2"/><!-- UniMRCP profile to use for ASR --><param name="default-asr-profile" value="unimrcpserver-mrcp2"/>

3）、设置拨号计划

在/usr/local/freeswitch/conf/dialplan/default.xml 文件中创建拨号计划：

```
<extension name="unimrcp">
    <condition field="destination_number" expression="^5001$">
        <action application="answer"/>
        <action application="lua" data="baidu.lua"/>
    </condition></extension>
```

3、话术业务逻辑的实现

1）、把对话内容通过 XML 配置起来，程序通过读取 XML 来跑你的对话

2）、每个对话以节点的形式进行，节点与节点之间的切换可以采用 JAVA 的状态设计模式进行切换，不懂这个设计模式可以去百度下

3）、在 eslEvent 里，通过 UUID 获取到你保存的状态对象，然后进行调用

4）、playtext 就是话术，trigger 就是对应的用户说的关键字，然后触发节点切换

5）、通过自定义 XML，结合状态设计模式跟策略模式进行节点切换，就变得很简单了

6）、但是如果要做成比较复杂的话，涉及到工作流

话术文件设置如下：

```
    type="regular", 通过正则表达式匹配
    type="number", 系统自动替换非数字字符后，强制使用正则表达式进行匹配，也就是type为number时，必须是采用正则表达式
-->
<nodes id="order" startRef="helloStrategy" endRef="successStrategy;endStrategy;endTrandStrategy">
    <node id="helloStrategy" nextRef="askPhoneStrategy">
        <playText>嗯，您好，很高兴为您服务</playText>
        <playWav></playWav>
        <asrErr></asrErr>
    </node>
    <!-- 表示如果没有命中相关节点，则重复执行当前节点三次，三次失败后进入对应的节点 -->
    <node id="startStrategy" tryMillSeconds="3000" tryCount="2" tryFailRef="endStrategy">
        <playText>请问您是${name}吗？</playText>
        <playWav></playWav>
        <asrErr></asrErr>
        <!-- 客户直接告诉姓氏，也直接进入地址确认流程 -->
        <trigger ref="confirmAddressStrategy" type="regular" keywords="(?&lt;=姓).*" putValue="name"/>
        <!-- 不是本人 -->
        <trigger ref="askNameStrategy" type="key" keywords="不是"/>
        <!-- 确认身份后进入地址确认流程 -->
        <trigger ref="confirmAddressStrategy" type="key" keywords="是的;对"/>
    </node>
    <!-- 表示1秒内没有识别出，则继续执行下一节点 -->
    <node id="askNameStrategy" tryMillSeconds="3000" tryCount="1" tryFailRef="confirmAddressStrategy">
        <playText>请问您的贵姓是什么呢？</playText>
        <playWav></playWav>
        <asrErr></asrErr>
        <!-- 回答姓氏 -->
        <trigger ref="confirmAddressStrategy" type="regular" keywords="(?&lt;=姓)\S$" putValue="name"/>
    </node>
    <!-- 确认地址流程 -->
    <node id="confirmAddressStrategy" tryMillSeconds="3000" tryCount="1" tryFailRef="endStrategy">
        <playText>请问您的下单地址是不是${address}</playText>
        <playWav></playWav>
        <asrErr></asrErr>
        <!-- 修改地址 -->
        <trigger ref="transToStrategy" type="key" keywords="不是;不对"/>
        <!-- 确认地址为上一个 -->
        <trigger ref="confirmPhoneStrategy" type="key" keywords="对;是"/>
    </node>
    <!-- 客户发件地址改变，则转接到人工 -->
    <node id="transToStrategy">
        <playText>请稍后，正在为您转回人工下单</playText>
        <playWav></playWav>
        <asrErr></asrErr>
    </node>
    <!-- 地址确认后，确认联系电话是多少 -->
    <node id="confirmPhoneStrategy" tryMillSeconds="5000" tryCount="2" tryFailRef="rightPhoneStrategy">
        <playText>请问联系电话是来电号码吗？</playText>
        <playWav></playWav>
        <!-- 修改联系电话 -->
        <trigger ref="askPhoneStrategy" type="key" keywords="不是;换一个"/>
        <!-- 确认联系电话 -->
        <trigger ref="successStrategy" type="key" keywords="对;是;没错"/>
    </node>
```

TTS播放

根据识别内容进行节点切换

Esl 核心处理代码：

```
String speech=null;
//如果开始策略不为空则开始开始策略
if(startStrategy!=null&&!startStrategy.equals("")) {
    //处理第一个开始策略
    Node startNode=nodeMaps.get(startStrategy);
    FsSocketAPI.startAsr(channel,startNode.getPlayText());
    try {
        Thread.sleep(8000);
    } catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    //处理第一个节点，并且得到下一个节点
    speech = ClientInfoManager.UUID_PLAYBACK_TEXT.get(uuid);

    if(speech==null){
        FsSocketAPI.playBack(channel,nodeMaps.get("endStrategy").getPlayText()
        try {
            Thread.sleep(8000);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return;
```

```
Node nodeNext=null;
List<Trigger> startNodeTriggers=startNode.getTriggers();
//遍历trigger进行一个循环
for (Trigger trigger : startNodeTriggers) {
    //将命中关键字的trigger中的节点赋值给nodeNext
    List<String> kList=trigger.getKeywords();
    //对关键词的一个验证，利用相似度算法进行一个比较
    Iterator<String> iterator=kList.iterator();
    while (iterator.hasNext()) {
        String keyword=iterator.next();
        if (speech.equals(keyword)) {
            nodeNext=nodeMaps.get(trigger.getRef());
            break;
            //这就一点定要保证不是，放在是前面
        } else if (speech.contains(keyword)||keyword.contains(speech)) {
            nodeNext=nodeMaps.get(trigger.getRef());
            break;
        } else if (SimilarityUtil.getSimilarityRatio(speech, keyword)>0.5) {
            nodeNext=nodeMaps.get(trigger.getRef());
            break;

        } else{
            nodeNext=nodeMaps.get(startNode.getTryFailRef());
        }
    }
```

4、呼叫中心的实现

1）设置呼叫计划

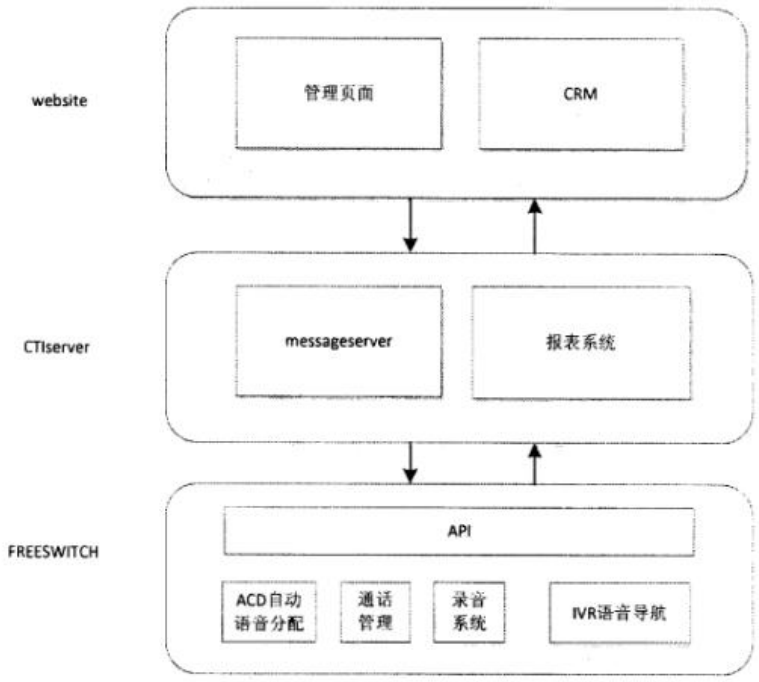
```
<!-- a sample IVR<action application="callcenter" data="support@default"/>  -->
<extension name="Socket Example">
  <condition field="destination_number" expression="^5006$">
    <action application="answer"/>
    <action application="playback" data="shout://tsn.baidu.com/text2audio?lan=zh&ctp=
    <action application="socket" data="192.168.3.248:8484 async full" />
  </condition>
</extension>

<!--<action application="answer"/>-->
<extension name="Callcenter Example">
  <condition field="destination_number" expression="^5007$">
    <action application="callcenter" data="support@default"/>
  </condition>
</extension>
```

2）设置座席队列

```
<!-- WARNING: Configuration of XML Agents will be updated into the DB upon restart. -->
<!-- WARNING: Configuration of XML Tiers will reset the level and position if those were
<!-- WARNING: Agents and Tiers XML config shouldn't be used in a multi FS shared DB setu
  <agents>
    <agent name="1003@default" type="callback" contact="[leg_timeout=10]user/1003" statu
  </agents>
  <tiers>
    <!-- If no level or position is provided, they will default to 1.  You should do thi
   <tier agent="1003@default" queue="support@default" level="1" position="1"/>
  </tiers>
```

5、后台业务逻辑的实现

使用 Spring Boot+REST 架构实现与前端和 Freeswitch 之间的交互。Freeswitch 可以加载 xml_cdr 模块，加载过后可以重新编译。这边后台的部分就是 CTIServer。



5.1)用户的注册登录

5.2)用户的通话记录

5.3)话术自动拨打

将选中的号码，通过 Freeswitch 拨打出去并实现 IVR 进行营销或者服务，然后再用 ASR 进行语音交互。

6、网页电话 webrtc 的实现

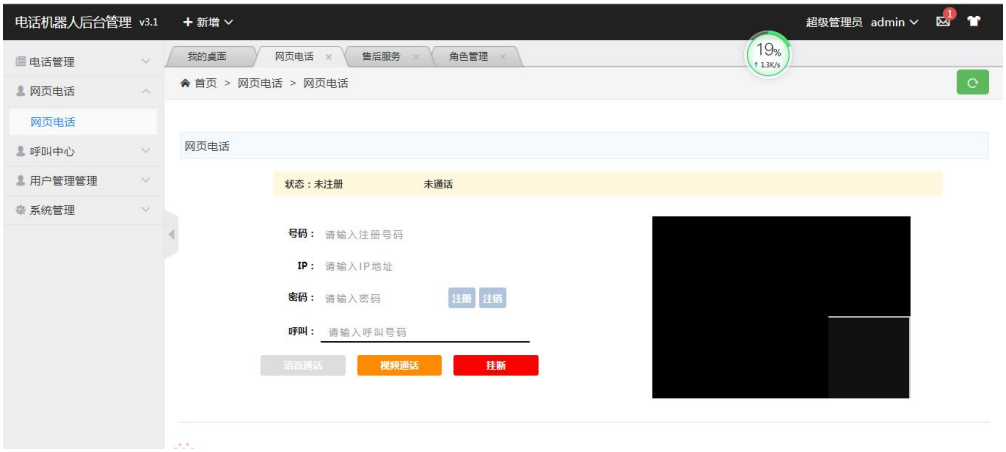
WebRTC，名称源自网页即时通信（英语：Web Real-Time Communication）的缩写，是一个支持网页浏览器进行实时语音对话或视频对话的 API。它于 2011 年 6 月 1 日开源并在 Google、Mozilla、Opera 支持下被纳入万维网联盟的 W3C 推荐标准。

WebRTC 实现了基于网页的视频会议，标准是 WHATWG 协议，目的是通过浏览器提供简单的 javascript 就可以达到实时通讯（Real-Time Communications (RTC)）能力。

WebRTC（Web Real-Time Communication）项目的最终目的主要是让 Web 开发者能够基于浏览器（Chrome\FireFox\...）轻易快捷开发出丰富的实时多媒体应用，而无需下载安装任何插件，Web 开发者也无需关注多媒体的数字信号处理过程，只需编写简单的 Javascript 程序即可实现，W3C 等组织正在制定 Javascript 标准 API，目前是 WebRTC 1.0 版本，Draft 状

态；另外 WebRTC 还希望能够建立一个多互联网浏览器间健壮的实时通信的平台，形成开发者与浏览器厂商良好的生态环境。同时，Google 也希望和致力于让 WebRTC 的技术成为 HTML5 标准之一，可见 Google 布局之深远。

WebRTC 提供了视频会议的核心技术，包括音视频的采集、编解码、网络传输、显示等功能，并且还支持跨平台：windows，linux，mac，android。



7、前端页面的实现

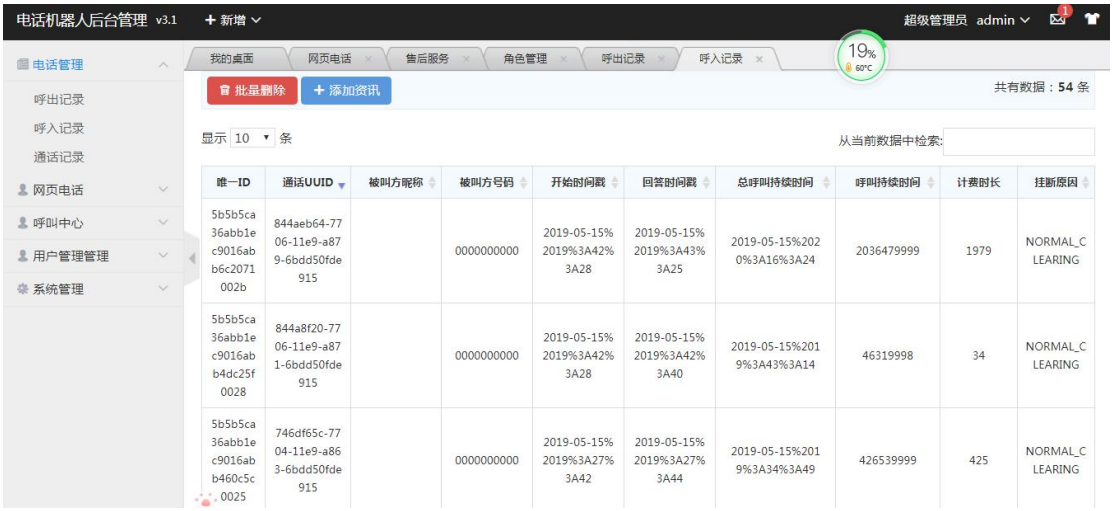
前端主要使用 jquery 中的 ajax 中的 get/post 和后台进行数据交互，数据的格式都是为 json。

1、登录注册

采用了 jquery.js，validate.js 实现登陆的验证码，背景采用了 css3.0 同时结合了 js 的动态效果。

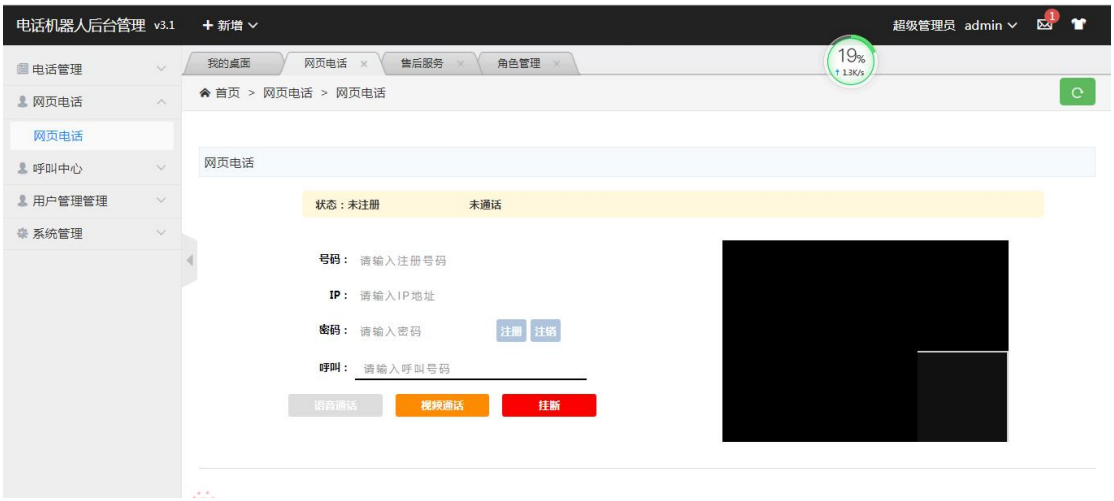
2、电话记录管理

针对呼入呼出进行分类，甚至还可以实现呼入录音，呼出录音。



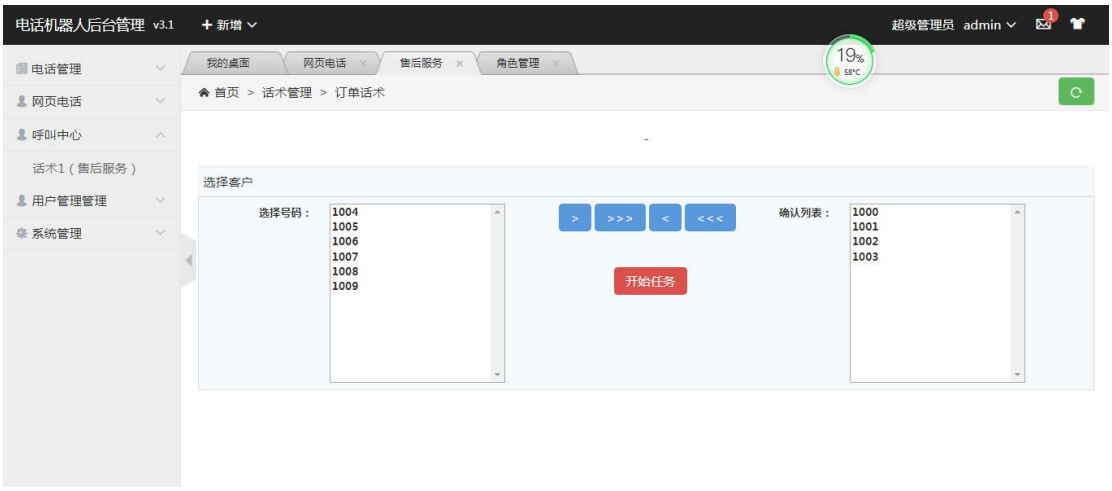
唯一ID	通话UUID	被叫方昵称	被叫方号码	开始时间戳	回答时间戳	总呼叫持续时间	呼叫持续时间	计费时长	挂断原因
5b5b5ca36abb1ec9016abb6c2071002b	844aeb64-7706-11e9-a879-6bdd50fde915		0000000000	2019-05-15%2019%3A42%3A28	2019-05-15%2019%3A43%3A25	2019-05-15%2020%3A16%3A24	2036479999	1979	NORMAL_CLEARING
5b5b5ca36abb1ec9016abb4dc25f0028	844a8f20-7706-11e9-a871-6bdd50fde915		0000000000	2019-05-15%2019%3A42%3A28	2019-05-15%2019%3A42%3A40	2019-05-15%2019%3A43%3A14	46319998	34	NORMAL_CLEARING
5b5b5ca36abb1ec9016abb460c5c0025	746df65c-7704-11e9-a863-6bdd50fde915		0000000000	2019-05-15%2019%3A27%3A42	2019-05-15%2019%3A27%3A44	2019-05-15%2019%3A34%3A49	426539999	425	NORMAL_CLEARING

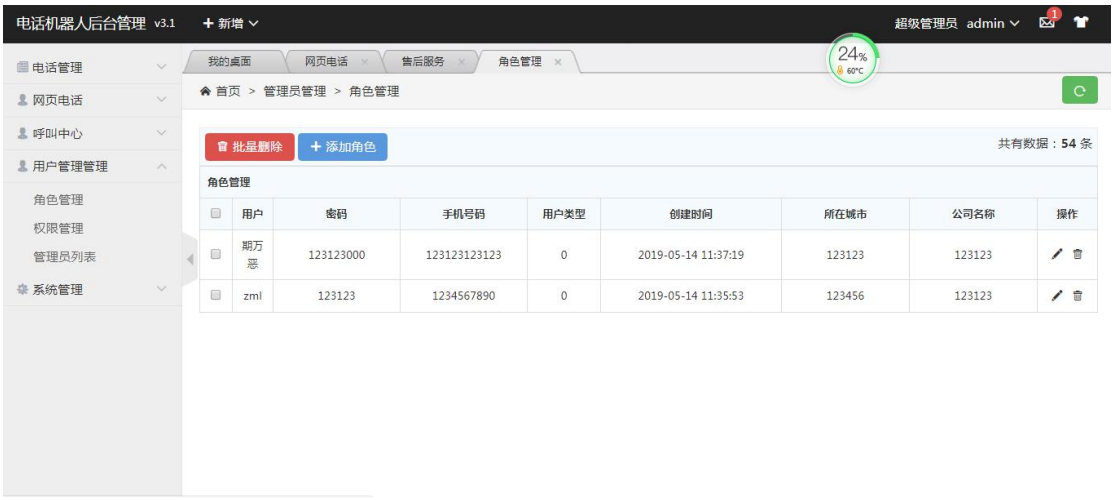
3、webrtc 网页电话



4、订单话术

拨打电话页面实现一个




电话机器人后台管理 v3.1
+ 新增
超级管理员 admin
1
我的桌面
网页电话
售后服务
角色管理
24%
60°C
电话管理
网页电话
呼叫中心
用户管理管理
角色管理
权限管理
管理员列表
系统管理
首页 > 管理员管理 > 角色管理
批量删除
+ 添加角色
共有数据：54 条
角色管理
用户
密码
手机号码
用户类型
创建时间
所在城市
公司名称
操作
期万恶
123123000
123123123123
0
2019-05-14 11:37:19
123123
123123
zml
123123
1234567890
0
2019-05-14 11:35:53
123456
123123